IP at Work: Fragments Assemble!
Packet's too big?
  Fragment, transmit and
  At receiving endpoint coalesce
If something goes awry?
  The network shrugs does nothing
If all fragments do get through?
  Coalesce with needlework so fine
  That the transport can't spot the seams

Frequently asked questions from the previous class survey
- Why does DHCP use two ports?
- Do closed networks only use the link layer?
- How do companies not end up manufacturing NICs with duplicate MAC addresses?
- What is included in a link layer frame's footer/trailer?
- How is data put back together?
- Are packets always the same size in WiFi networks?
- Why do data corruptions happen?
- Why don't personal computers have their own WiFi?
Topics covered in today’s lecture

- IPv4
- IPv6
Internet architecture

- Evolved out of experiences with ARPANET
  - Funded by ARPA of the US DoD
- A 4-level model

Internet protocol graph

End-to-End protocols

Ethernet  Fiber Distributed Data Interface

FTP  HTTP  NV  TFTP

TCP  UDP

IP

NET₁  NET₂  ...  NETₙ
Alternative view of the Internet Architecture

Internet architecture

- Does not imply strict layering
  - Bypassing immediate lower layers is possible

- Layer has an hour-glass shape
  - Wide at top and bottom
  - Narrow in the middle
  - IP is the focal point of the architecture
INTERNETWORKING

Internetwork

- Arbitrary collection of interconnected networks
  - To provide some sort of host-host packet delivery service

- Network of networks
  - Made up of lots of smaller networks
Internet Protocol (IP)

- Key tool to build scalable, **heterogeneous** networks
- Runs on all nodes (hosts and routers)
- Allows nodes and networks to **function as a single logical network**
- Possible to build an internetwork without IP
  - But IP is the only one that has faced **scale** issues
A simple internetwork: Communication between H1-H8

Example depicting how hosts (H1-H8) are logically connected
The IP service model

- Datagram model of **delivery**
  - Connectionless
  - Best effort
- **Addressing** scheme
  - Identifies all hosts in the internetwork

Datagram delivery

- Datagram is a type of packet
  - Sent in a **connectionless** fashion
- No need for any **advance** setup mechanisms
  - That tell network what do when packet arrives
- Every datagram contains enough information
  - To forward packet to correct destination
The network makes a best effort to send datagrams across

- Things that could go **wrong** with the packets
  - Lost
  - Corrupted
  - Misdelivered
  - Out of order and duplicates

- When things go wrong, the network does **nothing**
  - No attempt to recover from the failure

Keeping routers simple was one of the original design goals of IP

- Important to **run over anything**

- Putting extra functionality into routers to make up for network deficiencies?
  - Not a good idea

- Higher-level protocols/apps that run above IP need to be aware of failure modes
The IP Packet format consists of a header followed by bytes of data

- Represented as a succession of **32-bit** words
- Packet formats designed to align on 32-bit boundaries
  - Simplifies task of processing in software
- Transmission order
  - **Top word** transmitted first
  - **Leftmost byte** of each word transmitted first

---

The IPv4 packet header

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>HLen</td>
<td>TOS</td>
<td>Flags</td>
<td>Length</td>
<td>Offset</td>
</tr>
<tr>
<td>Ident</td>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td>SourceAddr</td>
<td>DestinationAddr</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>Pad (variable)</td>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IP Packet format

- **Version**
  - Makes it easy to redefine packet format later on

- **HLen**
  - Specifies length of header in 32-bit words
  - When there are no options (most of the time)
    - Header is 5 words or 20 bytes

- **TOS (type of service)**
  - Allow packets to be treated differently
    - Based on application needs

---

**Length**

- Length of the datagram in bytes
- Maximum size of IP datagram is $2^{16}$ bytes

**SECOND WORD OF IP PACKET**

- {Ident, Flags, Offset}
- Information about fragmentation
IP Packet format

- **TTL** (time to live)
  - Hop-count not timer (as originally intended)

- **Protocol** field
  - **Demultiplexing** key
    - Identifies higher-level protocol
    - TCP (6), UDP (17)

- **Checksum**
  - Consider IP header as a sequence of 16-bit words

---

IP Packet format

- **SourceAddr**
  - Decide if packet should be accepted
  - Also used for replies

- **DestinationAddr**
  - Full address of destination
  - Forwarding decisions are made at each router

- **Presence or absence of options**
  - Can be checked based on size of Hlen (without options header is 20 bytes)
IP Packet format

TOS field {Type of Service}

- Meant to specify how the datagram should be handled as it traversed the internet
  - Preference for low delay
  - Preference for high throughput
  - Preference for high reliability

- In practice TOS was not widely implemented

The 8 bits allocated to TOS can be divided into 5 parts

<table>
<thead>
<tr>
<th>Precedence bits</th>
<th>Indicates importance of datagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5</td>
<td></td>
</tr>
<tr>
<td>4 3 2</td>
<td>Low delay</td>
</tr>
<tr>
<td>1 0</td>
<td>High throughput</td>
</tr>
<tr>
<td></td>
<td>High reliability</td>
</tr>
</tbody>
</table>

Unused

7: Most Significant Bit
0: Least Significant Bit
Providing host-to-host service model over heterogeneous collection of networks

- Each network technology has its own idea of how large a packet can be
  - Ethernet v2: 1500 bytes
  - FDDI: 4352 bytes

Every network type has a Maximum Transmission Unit (MTU)

- Largest IP datagram that it can carry in its frame
- Smaller than the largest packet-size of network
  - IP datagram needs to fit in payload of link-layer frame
Ethernet Packet Layout

<table>
<thead>
<tr>
<th>Bytes:</th>
<th>7</th>
<th>1</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>46 &lt; Length &lt; 1500</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Preamble</td>
<td>S</td>
<td>Destination address</td>
<td>Source address</td>
<td>Length Of data</td>
<td>Data for Transmission</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

Preamble bit pattern: 10101010  Single Byte start ($) frame delimiter: 10101011

Fragmentation is necessary when datagram path includes network with smaller MTU

- All fragments carry same identifier in **Ident** field
  - To enable fragment reassembly
  - Chosen by the source host
- If all fragments do not arrive at receiving host?
  1. Receiver **gives up** reassembly [reassembly timeout: 15 seconds RFC0791]
  2. **Discards** fragments that did arrive
- IP **does not attempt** to recover from missing fragments
A simple internetwork: Sending IP datagrams from H1 to H8

IP datagrams traversing a sequence of physical networks
IPv4 Packet header

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>HLen</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>TOS</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>Length</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Ident</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>Flags</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>Offset</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>TTL</td>
<td>28</td>
<td>32</td>
</tr>
<tr>
<td>Protocol</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>Checksum</td>
<td>36</td>
<td>40</td>
</tr>
<tr>
<td>SourceAddr</td>
<td>40</td>
<td>44</td>
</tr>
<tr>
<td>DestinationAddr</td>
<td>44</td>
<td>48</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>48</td>
<td>52</td>
</tr>
<tr>
<td>Pad (variable)</td>
<td>52</td>
<td>56</td>
</tr>
<tr>
<td>Data</td>
<td>56</td>
<td>60</td>
</tr>
</tbody>
</table>

Header fields used in IP fragmentation:
Fragmentation occurs at 8-byte boundaries

Unfragmented packet

Fragmented packet
Header fields used in IP fragmentation:

Fragmentation occurs at **8-byte boundaries**

<table>
<thead>
<tr>
<th>Start of header</th>
<th></th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ident = x</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>Rest of header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 data bytes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fragmented packet

<table>
<thead>
<tr>
<th>Start of header</th>
<th></th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ident = x</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>Rest of header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>376 data bytes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IPv6 (and comparing with IPv4)**
IPv6 versus IPv4: Key Differences

- Source and destination addresses are 128-bits (16 bytes) in IPv6.
- IPv6 treats Options as extension headers.
- To simplify processing of packets in routers, IPv6 did away with fragmentation.
  - Responsibility for packet fragmentation is at the end points.
  - IPv6 hosts must perform: (1) path MTU discovery, (2) perform end-to-end fragmentation, OR (3) send packets no larger than the default MTU=1280.
- As of 2014, IPv4 still carried >99% of worldwide Internet traffic.
- In 2022, Google reported IPv6 access reaching ~40%.

IPv6 Packet Header

IPv6 Packet Header is fixed at 40 bytes ... So there is no Header Length.
IPv6 Packet Header: Some more details [1/2]

- **Version**: 4 bits [0110]
- **Traffic Class**: 6+2 bits
  - Differentiated Services for QoS
  - Anything that ends in 2 "1" bits is intended for experimental or local use
- **Flow Label (20 bits)**
  - If it is non-zero: Serves as a hint to routers and switches with multiple outbound paths that these packets should stay on the same path, so that they will not be reordered
- **Payload length (16 bits)**: Size of payload including extension headers

IPv6 Packet Header: Some more details [2/2]

- **Next Header (8 bits)**
  - Specifies the type of the next header
- **Hop Limit (8 bits)**
  - Replaces the time-to-live field of IPv4
- **Destination and Source Addresses (128-bits or 16 bytes each)**
  - The mass of earth is $2^{92}$ grams
- **Note**: The IPv6 packet **header has no checksum**
  - Transport or application layer protocols are assumed to provide sufficient error detection
**Structure of the IPv6 Packet**

PDU typically contains an upper layer protocol header and its payload. For e.g.: a TCP segment, UDP Datagram, or an ICMPv6 message.

---

**Extension Header**

- If the Next Header field is non-zero
  - It defines an extension header
- Current extension header types
  - Information for routers, route definition, fragment handling, authentication, encryption, etc.
- Each extension header has a specific size and defined format
Extension Header

- If an extension header is present?
  - Follows the basic header and precedes the payload AND
  - Includes a Next Header

- Every extension header starts off with the Next Header

IPv6 Extension Headers: The chain of pointers using the Next Header field

Each extension header must fall on a 64-bit (8-byte) boundary. Use Padding to get there if less than that.
**UDP**

**Simple Demultiplexer**

User Datagram Protocol

- **Simplest** possible transport protocol
  - Extends host-to-host into process-to-process communications

- No additional functionality to best-effort service provided by underlying network

- Adds **demultiplexing**
  - Allows applications on a host to **share** the service
UDP identification of processes

- Processes *indirectly* identify each other
  - Abstract locator called *port*

- Source sends a message to a port
  - Destination receives messages from a port

- Process is identified by a *port on a particular host*

Format of a UDP header

<table>
<thead>
<tr>
<th>0</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>DstPort</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
A port is just an abstraction

- Typically implemented as a **message queue**
- When message arrives?
  - Protocol appends message to end of the queue

- **UDP**
  - If the queue is full, message is discarded
  - No flow-control mechanism

UDP message queue: The port abstraction

![Diagram showing packet demultiplexing and process connections through a UDP port]

- Packets arrive
- Packets demultiplexed
- Ports to Application Processes
Some work that UDP does do besides demultiplexing:
Checksumming

- UDP header
- Message body
- Pseudoheader: From the IP header
  - Protocol number
  - Source IP address
  - Destination IP address
- UDP length
  - Used twice

Verify if message is delivered between the correct endpoints

The contents of this slide-set are based on the following references